Dynamic-Net: Tuning the Objective Without Re-training for Synthesis Tasks

Alon Shoshan
Technion, Israel
shoshan@campus.technion.ac.il

Roey Mechrez
Technion, Israel
roey@campus.technion.ac.il

Lihi Zelnik-Manor
Technion & Alibaba Group
lihi@technion.ac.il

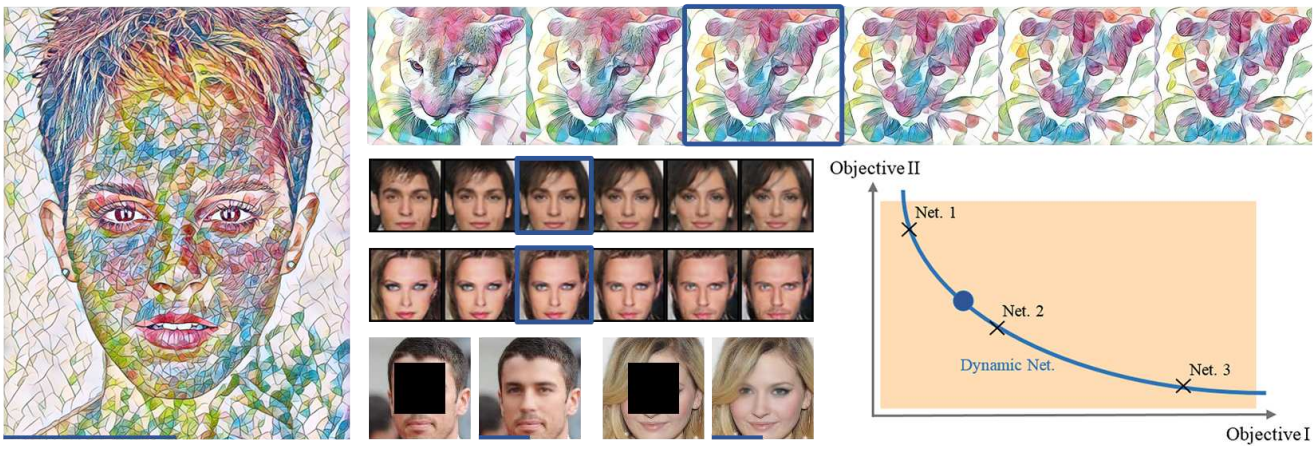


Figure 1: **Dynamic-Net:** We propose an approach that enables traversing the "objective-space", spanned by two different objectives, at test-time, without re-training, as illustrated by the blue dot moving along the blue curve in the plot. This is different from the common practice of training a separate network for each objective, represented by $\times$'s on the plot. Using a single Dynamic-Net we can tune the level of stylization of an image, monitor completion quality per image, or control facial attributes, all interactively at test-time, without re-training. [Animated figure, please view at our webpage].

Abstract

One of the key ingredients for successful optimization of modern CNNs is identifying a suitable objective. To date, the objective is fixed a-priori at training time, and any variation to it requires re-training a new network. In this paper we present a first attempt at alleviating the need for re-training. Rather than fixing the network at training time, we train a "Dynamic-Net" that can be modified at inference time. Our approach considers an "objective-space" as the space of all linear combinations of two objectives, and the Dynamic-Net is emulating the traversing of this objective-space at test-time, without any further training. We show that this upgrades pre-trained networks by providing an out-of-learning extension, while maintaining the performance quality. The solution we propose is fast and allows a user to interactively modify the network, in real-time, in order to obtain the result he/she desires. We show the benefits of such an approach via several different applications.

1. Introduction

"I can't change the direction of the wind, but I can adjust my sails to always reach my destination"

—Jimmy Dean

A common practice in image generation is to train a deep network with an appropriate objective. The objective is often complex and integrates multiple loss terms, e.g., in style transfer [6, 11], super-resolution [2, 18], inpainting [16], image-to-image transformations [10], domain transfer [19, 12] and attribute manipulation [24]. To date, the choice of the specific objective, and the trade-off between its multiple terms, are set a-priori during training. This results in trained networks that are fixed for a specific working point. This is limiting for three reasons. First, oftentimes one would like the flexibility to produce different results, e.g., stronger or weaker style transfer. Second, in many cases the best working point is different for different inputs. Last, it is hard to predict the optimal working point, especially when the full objective is complex and when ad-

versarial training [7] is incorporated. Therefore, practitioners perform greedy search over the space of objectives during training, which demands significant compute time.

In this paper we propose an alternative approach, called *Dynamic-Net*, that resolves this for some scenarios. Rather than training a single fixed network, we split the training into two phases. In the first, we train the blocks of a "main network" using a certain objective. At the second phase we train additional residual [8] "tuning-blocks", using a different objective. Then, at inference time, we can decide whether we want to incorporate the tuning-blocks or not and even control their contribution. This way, we actually have at hand a dynamic network that can be assembled at inference time from the main network and tuning-blocks. Our underlying assumption is that the tuning-blocks can capture the variation between the two objectives, thus allowing traversal of the objective space. The Dynamic-Net can thus be easily geared towards the first or second objective, by tuning scalar parameters, at test-time.

The key idea behind our approach is inspired by the Jimmy Dean citation at the beginning of the introduction. We acknowledge that we cannot directly modify the objective at test-time. However, what we can do is modify the latent space representation. Therefore, our approach relies on manipulation of deep features in order to emulate a manipulation in objective space.

The main advantages of the Dynamic-Net are three-fold. First, using a single training session the Dynamic-Net can emulate networks trained with a variety of different objectives, for example, networks which produce stronger or weaker stylization effects, as illustrated in Figure 1. Second, it facilitates image-specific and user-specific adaptation, without re-training. Via a simple interface, a user can interactively choose at real-time the level of stylization or a preferred inpainting result. Last, the ability to traverse the objective space at test-time shrinks the search-space during training. More specifically, we show that even when the choice of objective for training is sub-optimal, the Dynamic-Net can reach a better working point.

We show these benefits through a broad range of applications in image generation, manipulation and reconstruction. We explore a variety of objectives and architectures, and present both qualitative and quantitative evaluation. Our code is available at https://github.com/AlonShoshan10/dynamic_net.

2. Related Work

Multi-loss objectives Many state-of-the-art solutions for image manipulation, generation and reconstruction utilize a multi-loss objective. For example, Isola et al. [10] combine $L1$ and adversarial loss [7] for image-to-image transformation. Johnson et al. [11] trade-off between a style loss (i.e. Gram loss) and content loss (i.e. the perceptual loss) for fast style-transfer and super-resolution, SRGAN [13] balances between a content loss and adversarial loss for perceptual super-resolution, and [26] combines $L1$, a style loss and an adversarial loss for texture synthesis. In all of these cases, the weighting between the loss terms is fixed during training, producing a trained network that operates at a specific working point.

The impact of the trade-off between multiple objectives has been discussed before in several contexts. [2] show that in image restoration algorithms there is an inherent trade-off between distortion and perceptual quality. They analyze the trade-off and show the benefits of different working points. In [6] it is shown empirically that a different balance between the style loss and content loss leads to different stylization effects.

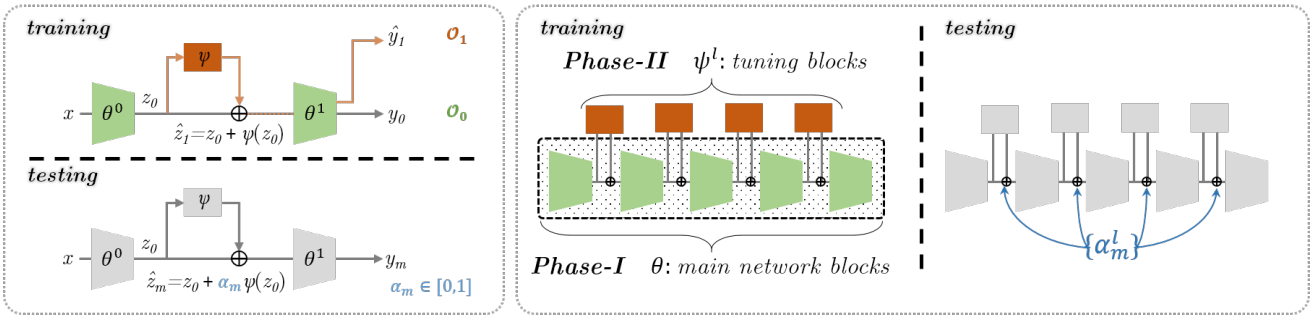
The importance and difficulty of choosing the optimal balance between different loss terms is tackled by methods for multi-task learning [4, 20, 14, 23, 22]. In these works a variety of solutions have been proposed for learning the weights for different tasks. Mutual to all of these methods is that their outcome is a network trained with a certain fixed balance between the objectives.

Deep feature manipulation The approach we propose is based on training "tuning-blocks" that learn how to manipulate deep features in order to achieve a certain balance between the multiple objectives. It is thus related to methods that employ manipulation of deep features in latent space. These methods are based on the basic hypothesis of [1] that "CNNs linearize the manifold of natural images into a Euclidean subspace of deep features", suggesting that linear interpolation of deep features makes sense. Inspired by this hypothesis, [25] learn the linear "direction" that modifies facial attributes, such as adding glasses or a mustache. In [3] a more sophisticated manipulation approach is proposed. They introduced blocks to be added to an auto-encoder network in order to learn the required manipulation to modify a facial attribute. While producing great results on manipulation of face images, their approach implicitly assumes that the training images are similar and roughly aligned.

3. Proposed Approach: Dynamic-Net

To date, one has to re-train the network for each objective. In this section we propose Dynamic-Net that allows changing the objective at inference time, without re-training. Dynamic-Nets can emulate a plethora of "intermediate working points" between two given objectives $\mathcal{O}_0$ and $\mathcal{O}_1$, by simply tuning a single parameter. One can think of this as implicit interpolation between the objectives. Our solution is relevant in cases where such an interpolation between the objectives $\mathcal{O}_0$ and $\mathcal{O}_1$ is meaningful.

To provide some intuition we begin with an example. A common scenario where an "intermediate working point" is



(a) Single-block framework (b) Multi-block framework

Figure 2: **Proposed framework:** Our training has two steps: (i) First the "main" network θ (green blocks) is trained to minimize $\mathcal{O}_0$. (ii) Then θ is fixed, one or more tuning-blocks ψ are added (orange block), and trained to minimize $\mathcal{O}_1$. The output $\hat{y}_1$ approximates the output y_1 one would get from training the main network θ with objective $\mathcal{O}_1$. At test-time, we can emulate results equivalent to a network trained with objective $\mathcal{O}_m$ by tuning the parameter α_m (in blue) that determines the latent representation z_m. Our method can be applied as (a) a single-block framework or as (b) multi-block framework.

intuitive, is when the objectives consist of a super-position of two loss terms: $\mathcal{O}_0 = \mathcal{L}_\mathcal{A} + \lambda_0 \mathcal{L}_\mathcal{B}$ and $\mathcal{O}_1 = \mathcal{L}_\mathcal{A} + \lambda_1 \mathcal{L}_\mathcal{B}$, where $\mathcal{L}_\mathcal{A}$, $\mathcal{L}_\mathcal{B}$ are loss terms, and λ_0, λ_1 are scalars. Assuming, without loss of generality, that $\lambda_0 \leq \lambda_1$, an intermediate working point corresponds to an objective $\mathcal{O}_m = \mathcal{L}_\mathcal{A} + \lambda_m \mathcal{L}_\mathcal{B}$, such that $\lambda_0 \leq \lambda_m \leq \lambda_1$. Our goal is to approximate at inference time the results of a network trained with any objective $\mathcal{O}_m$, while using only $\mathcal{O}_0$ and $\mathcal{O}_1$ during training.

The key idea behind the approach we propose, is to use interpolation in latent space in order to approximate the intermediate objectives. For simplicity of presentation we start with a simple setup that uses linear interpolation, at a single layer of the network. Later on we extend to non-linear interpolation.

3.1. Single-block Dynamic-Net

Our single-block framework is illustrated in Figure 2(a). It first trains a CNN, to which we refer as the "*main network blocks*", with objective $\mathcal{O}_0$. We then add an additional block to the network, to which we refer as the "*tuning-block*" ψ that learns the "direction of change" in latent space z, that corresponds to shifting the objective from $\mathcal{O}_0$ to another working point $\mathcal{O}_1$. Our hypothesis is that walking along the "direction of change" in latent space can emulate a plethora of "intermediate" working points $\mathcal{O}_m$ between $\mathcal{O}_0$ and $\mathcal{O}_1$.

In further detail, our pipeline is as follows:

▷ Training:

- Train the main network blocks by setting the objective to $\mathcal{O}_0$.
- Fix the values of the main network, add a tuning-block ψ between layers l and $l+1$, and post-train only ψ by setting the objective to $\mathcal{O}_1$. The block ψ will capture the variation between the latent representations z_0 and z_1, that correspond to $\mathcal{O}_0$ and $\mathcal{O}_1$, respectively.

▷ Testing:

Fix both the main blocks as well as the tuning block ψ, and do as follows:

- Propagate the input until layer l of the main network to get z_0.
- Generate an "intermediate" point in latent space, $z_m = z_0 + \alpha_m \psi(z_0)$, by tuning the scalar parameter α_m.
- Propagate z_m through the rest of the main network to obtain outcome y_m that corresponds to objective $\mathcal{O}_m$.

The justification for our approach stems from the following two assumptions:

Assumption 1 *We adopt the hypothesis of [1] that "CNNs linearize the manifold of natural images into a Euclidean subspace of deep features".*

This assumption implies that the latent representation of an intermediate point can be written as $z_m = z_0 + \alpha_m(z_1 - z_0)$ where $\alpha_m \in [0, 1]$. Setting $\alpha_m = 0$ yields working point 0 while setting $\alpha_m = 1$ yields working point 1.

Assumption 2 *For any pair of working points $\mathcal{O}_0$,$\mathcal{O}_1$, with corresponding latent representations z_0, z_1, it is possible to train a block ψ such that $z_1 \approx z_0 + \psi(z_0)$.*

Putting assumptions 1 and 2 together suggests that we can approximate any intermediate working point $\mathcal{O}_m$ by computing $\hat{z}_m = z_0 + \alpha_m \psi(z_0)$ and have that $z_m \approx \hat{z}_m$.

To provide further intuition we revisit the example where the objectives are of the form $\mathcal{O} = \mathcal{L}_\mathcal{A} + \lambda \mathcal{L}_\mathcal{B}$. Here the parameter λ controls the balance between the two loss terms

$\mathcal{L}_{\mathcal{A}}$ and $\mathcal{L}_{\mathcal{B}}$. To interpolate in objective space we would like to modify λ but this is not possible to do directly at test-time. Instead, our scheme enables interpolation in latent space by modifying the parameter α, which controls z_m. Our main hypothesis is that the suggested training scheme will lead to a proportional relation between α and λ. That is, increasing α will correspond to a monotonic increase in λ, thus implicitly achieving the desired interpolation in objective space.

In the more general case, when the objectives $\mathcal{O}_0$ and $\mathcal{O}_1$ are of different forms, the interpolation we propose in objective space cannot be formulated mathematically so intuitively. Nonetheless, the conceptual meaning of such interpolation could be sensible. For example, we could train an image generation network with two different adversarial objectives, one that prefers blond hair and another that prefers dark hair. Interpolating between the two objectives should correspond to generating images with varying hair shades. Therefore, to prove broad applicability of the proposed approach to a variety of objectives, we present in Section 4 several applications and corresponding results.

3.2. Multi-block Dynamic-Net

In practice, adding a single tuning-block, at a specific layer, might be insufficient. It limits the manipulation to linear transformations in a single layer of the latent space. Therefore, we propose adding multiple blocks, at different layers of the network as illustrated in Figure 2(b).

The training framework is similar to that of single-block, except that now we have multiple tuning blocks ψ^l, each associated with a corresponding weight α^l_m. When training the tuning-blocks we fix all the weights to $\alpha^l_m = 1$. Then at inference-time, we can tune each of the weights independently to yield a plethora of networks and results.

4. Experiments

In this section we present experiments with several applications that demonstrate the utility of the proposed Dynamic-Net and support the validity of our hypotheses. In order to emphasize broad applicability we selected applications of varying nature, with a variety of loss functions and network architectures, as summarized in Table 1. Tuning-blocks were implemented as $conv$–$relu$–$conv$–$relu$–$conv$. Further implementation details, architectures and parameter values are listed in the supplementary.

The motivation behind Dynamic-Net was three-fold: (i) provide ability to modify the working point at test-time, (ii) allow image-specific adaptation, and (iii) reduce the dependence on optimal objective selection at training time. In what follows we explore these contributions one by one, through various applications.

In the next subsections, if not stated otherwise, we used the multi-block framework while setting all $\{\alpha^l\}$ to be equal, i.e., $\alpha^0 = \alpha^1 ... \equiv \alpha$, and we tune α.

Application	Objectives	Architecture
Style Transfer	$\mathcal{L}_{content}, \mathcal{L}_{style}$	[11]
Image Completion	$\mathcal{L}_{L1}, \mathcal{L}_{adv}$	[10]
Face Generation	$\mathcal{L}_{adv}$	[25]

Table 1: **Applications summary:** We evaluate Dynamic-Net on three different applications: image manipulation (style transfer), reconstruction (image completion) and generation (faces). The applications minimize different loss terms and are based on a variety of architectures.

4.1. Tuning the objective at test-time :: Style Transfer

Our first step is to show that the proposed approach can indeed traverse the objective space, and emulate multiple meaningful working points at test-time, without re-training. We chose to show this via experiments in Style Transfer.

Super-position of objectives: We begin with the common scenario where the objective-space is a super-position of two loss terms. We followed the setup of *fast style transfer* [11], where the goal is to transfer the style of a specific style image to any input image. This is done by training a CNN to optimize the objective: $\mathcal{O} = \mathcal{L}_{content} + \lambda \mathcal{L}_{style}$, where $\mathcal{L}_{content}$ is the Perceptual loss [6] between the output image and input image, and $\mathcal{L}_{style}$ is the Gram loss [6] between the output image and style image. The hyper-parameter λ balances between preserving the content image and transferring the texture and appearance of the style image. Our goal here is to show that tuning α of the Dynamic-Net at test-time can replace tuning of λ at training-time.

Following the training procedure suggested in Section 3 we first train the main network with objective $\mathcal{O}_0 = \mathcal{L}_{content} + \lambda_0 \mathcal{L}_{style}$, then freeze their weights and train the tuning-blocks with $\mathcal{O}_1 = \mathcal{L}_{content} + \lambda_1 \mathcal{L}_{style}$. Similar to [11] we use the MS-COCO [15] dataset for training.

Figure 3 shows a few example results together with the corresponding working points in the objective-space, which trade-offs the content and style loss terms. We successfully control the level of stylization, at test-time, by tuning α. An important result is that the working points emulated by the Dynamic-Net correspond to fixed networks trained for that specific working point (marked by $\times$) in terms of style loss and content loss.

The figure also compares to interpolation in image space, i.e., blending images produced by different fixed networks directly. For this baseline we use the following two networks; the main network of our Dynamic-Net and the closest fixed network (in terms of loss) to Dynamic-Net with

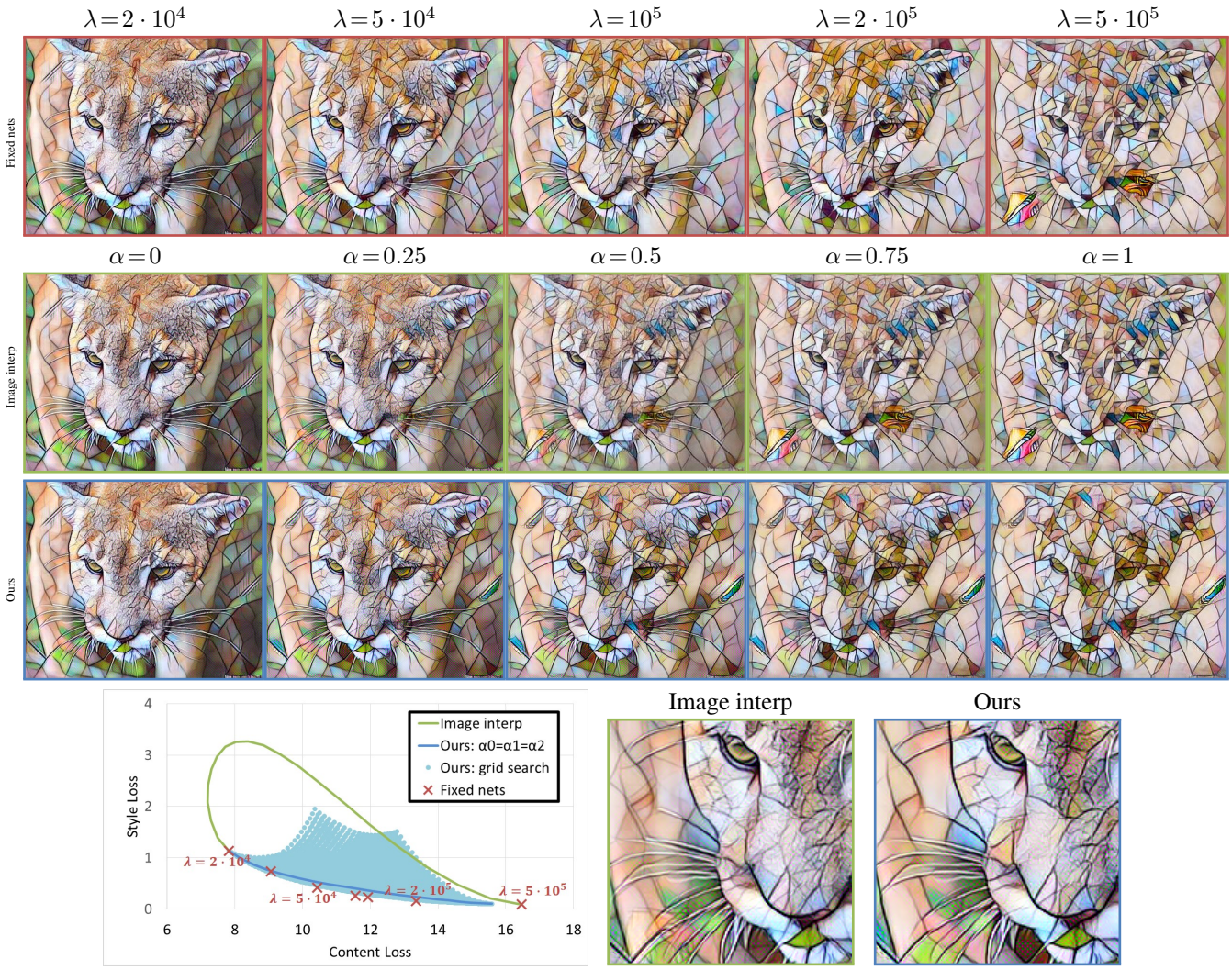


Figure 3: **Tuning the objective at test-time:** First row shows the results of the fixed networks, each was trained separately for a different objective (corresponding to the red $\times$'s). Second row shows results for image interpolation between two fixed nets - $\lambda = 2 \cdot 10^4$ and $\lambda = 5 \cdot 10^5$ (corresponding to the green curve) as baseline. Third row shows results of Dynamic-Net with three tuning-blocks (main network was trained with $\lambda_0 = 2 \cdot 10^4$ and tuning-blocks with $\lambda_0 = 10^6$) where we increase α from 0 to 1 and $\alpha = \alpha_0 = \alpha_1 = \alpha_2$ (corresponding to the blue curve). The graph shows the $\mathcal{L}_{style}$ vs. $\mathcal{L}_{content}$ where the cyan dots represent a grid search of Dynamic-Net for 1000 possible values of α_0, α_1, α_2. In the bottom right corner is a zoomed-in patch of our approach vs. image interpolation for $\alpha = 0.25$, it can be observed that the baseline is dissolving one image upon the other reader than naturally increase the style as our approach and empirical evidence can be seen in the graph.

$\alpha = 1$. It can be seen that the results are inferior qualitatively and quantitatively, since the style loss does not change monotonically.

Our method also allow tuning each tuning-block individually as can be observed by the grid search in the graph. Each point of the grid search represent a result produced with a different value of α_0, α_1 and α_2. This allows us to traverse the objective space in many interesting ways and even produce different images for the same working point.

Disjoint objectives: To further explore the generality of our approach we next experiment with disjoint objectives. As a case study we chose to traverse between stylization with two different style images. That is, $\mathcal{O}_0$ was trained with one style image, while $\mathcal{O}_1$ was trained with a different style image. At test time we tune α to traverse between the two objectives. Figure 4 presents results when the style images are completely different. We compare our result to two algorithms, Arbitrary Style Transfer using AdaIN [9] and Conditional-IN [5]. A third baseline is a simple interpola-

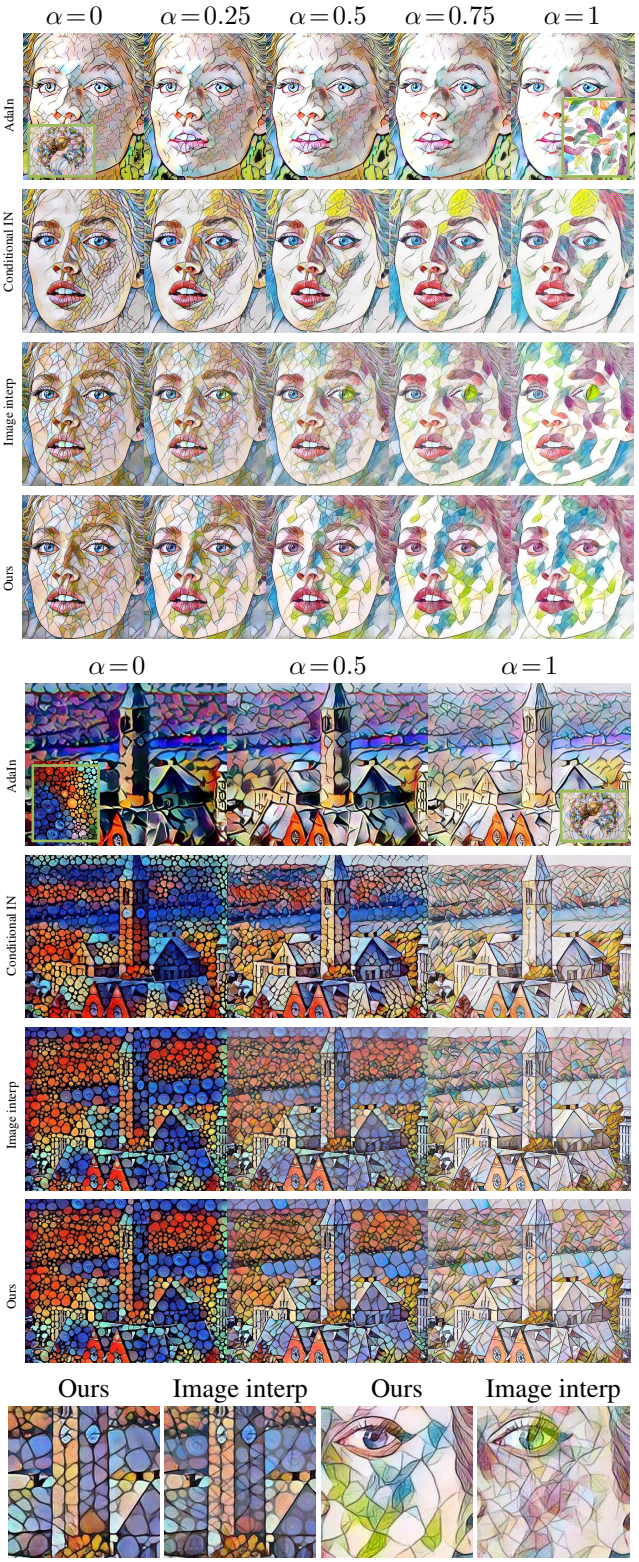


Figure 4: **Traversing between styles:** results of four different methods. Last row shows a zoomed-in patch of our method vs. image interpolation for $\alpha = 0.5$. Best viewed zoomed-in.

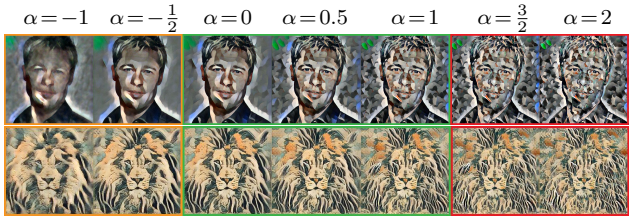


Figure 5: **Objective extrapolation:** Style transfer results where the main-blocks are trained with a *high resolution* style image, while the tuning-blocks are trained with a *low resolution* style image. Since the tuning-blocks capture the trend between the two style images the Dynamic-Net can generate different scales of texture: (green-box) interpolation along the style scale (red-box): extrapolation to the low-resolution side, and (yellow): extrapolation to the high-resolution side (style images in the supplementary).

tion in image space between results of two fixed networks. Conditional-IN reduces each style image into a point in an embedding space, each style shears the same convolutional weights of the network but it has its own normalization parameters. A blending effect is achieved by interpolating the normalization parameters of two styles. Arbitrary Style Transfer network consists of a fixed-encoder, AdaIN layer and a decoder. AdaIN is used to adjust the mean and variance of the content input image to match those of an arbitrary style image. Interpolating between AdaIN parameters of two styles produces a blending effect. For AdaIN we use the official implementation and pre-trained network, while for Conditional-IN we use the official implementation but trained the network for 10 different styles used in our paper. For the image interpolation baseline we use the following two networks; the main network of our Dynamic-Net and a fixed network trained for the second style image. Note that both AdaIN and Conditional-IN require specific constraints on the architecture, while our method is general and can extend existing high-quality pre-trained networks (as main networks). This can explain why our results are more faithful to the given style images. In addition, as we can observe from the zoomed-in images, our method achieves a natural blending of the two styles as opposed to a "fade away" effect of one image on top of another, formed by the baseline.

In Figure 5 the style images are two versions of the same style image, albeit at different resolutions. It can be seen that Dynamic-Net provides a smooth transition between the objectives. We also examine the ability to *extrapolate* in objective-space as shown in Figure 5. Specifically, we wanted to see if we can emulate working points that are not intermediate to those used during training. Interestingly, setting $\alpha < 0$ or $\alpha > 1$ also leads to meaningful results, corresponding to extrapolation in scale space of the style.

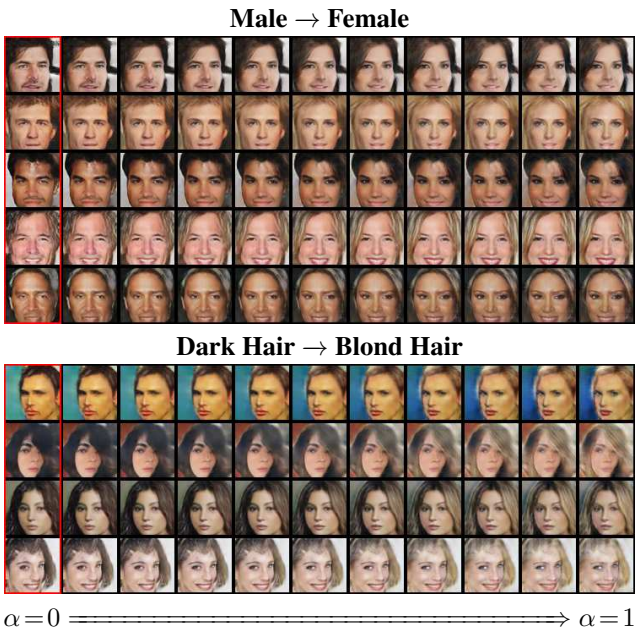


Figure 6: **Controlled generation results:** the proposed method allow us to interpolate between different facial attributes. The values of α are gradually increasing from left to right, results in a monotonic change of the specific attribute. Most left: $\alpha=0$ correspond to the baseline result of DCGAN [21].

4.2. The objective is user specific :: Face Generation

In some applications the desired output is not only image dependent but further depends on the user's preference. This could be observed previously in the style transfer experiments, were every user could prefer different stylization options. As another example for such a case we chose the task of face generation, where our approach endows the user with fine control over certain facial attributes, such as hair color or gender.

We adopted the architecture of DCGAN [21] that is trained with a single adversarial loss $\mathcal{O} = \mathcal{L}_{adv}$ over the CelebA [17] dataset. To provide control over an attribute, such as hair color, we split the dataset into two sub-sets, e.g., dark hair vs. blond. Both, the main network and the tuning-blocks were trained with an adversarial loss, but with different data sub-set. The two objectives are thus disjoint in this case.

At test-time, the user can tune α to generate a face with desired properties. For example, the user can tune the hair color or the masculinity of the generated face. Qualitative results are presented in Figure 6 for two attributes: male-to-female and dark hair-to-blond hair. It can be seen that our Dynamic-Net smoothly traverses between the two objectives, generating plausible images, with a smooth attribute control.

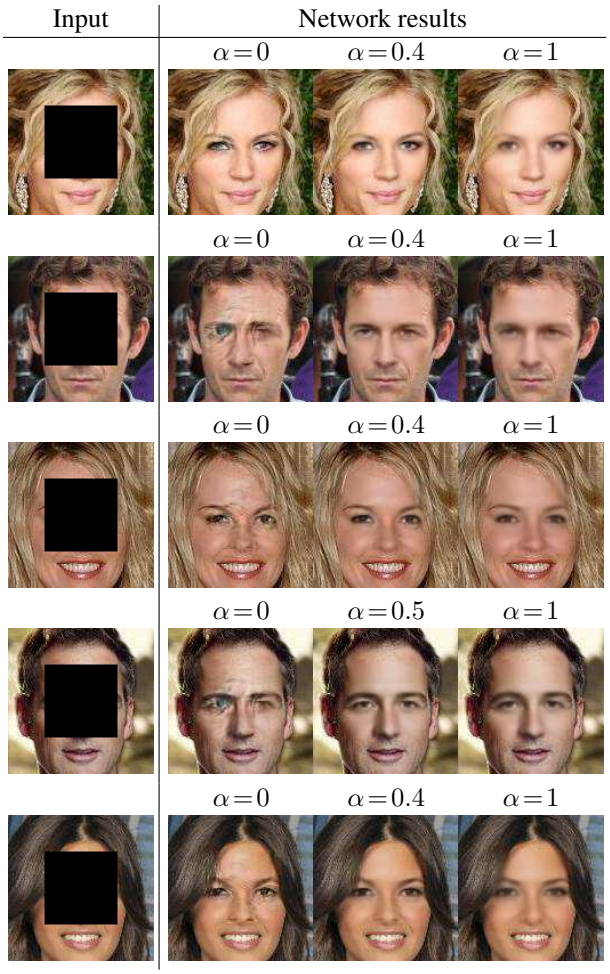


Figure 7: **Robustness to hyper-parameter:** The main network ($\alpha = 0$) produces artifacts common to adversarial training while $\alpha = 1$ produces blurry images common to L1 loss. Using $0 < \alpha < 1$ results in high quality images preventing the need to retrain the main network numerous times with different objectives to achieve high quality results.

4.3. Robustness to hyper-parameter :: Image Completion

Our last goal is to show that our approach shrinks the required search space over the objective at training time. In this experiment we use the task of image completion to show that by using tuning-blocks we can effectively prevent exhaustive re-training for tuning the objective. We intentionally train the main network for a sub-optimal objective that leads to poor quality completion and artifacts and use the tuning-blocks to adjust the objective post training to achieve high quality results. This was in order to show that even when the main network is of poor quality, adding the

tuning-blocks with an appropriate α could result in a better overall network, getting rid of the artifacts. This is possible because we can traverse the objective space and thus identify good working points, even when those used for training were sub-optimal.

In our experimental setup the input is a face image with a large hole at the center, and the goal is to complete the missing details in a faithful and realistic manner. As architecture we adopted a version of *pix2pix* [10] (see supplementary for details). The objective for training the main network was $\mathcal{O}_0 = \mathcal{L}_{L1} + \lambda\mathcal{L}_{adv}$ ($\lambda = 0.005$) and three tuning-blocks were trained with $\mathcal{O}_1 = \mathcal{L}_{L1}$.

Figure 7 shows some of our results. The main network ($\alpha = 0$) produces artifacts common for adversarial losses while on the other hand using the whole wight of the tuning-blocks ($\alpha = 1$) results in blurry images common to L1 losses. Using $0 < \alpha < 1$ We traverse the objective space and produce high quality images. This suggests that during training rather then trying multiple values for λ one can just select a single value, and then at test-time adapt α. The training of the tuning-blocks demonstrate robustness and implies that our Dynamic-Net forms a good alternative to the traditional greedy search. Choosing α can be done interactively in real-time, to tailor the network for a specific image. Since for different images there can be found a better objective that suit them specifically, interactively editing the results per image can be significant and difficult to achieve using fixed networks. Setting α is fast and provides an interesting alternative to hyper-parameter search at training time, both in terms of computing efficiency and as it enables image and user specific tuning.

5. Method Analysis

Limitations Figure 8 present the limitation of the proposed method, when using extreme objectives. Specifically, we trained the tuning-blocks *without* a style loss term, i.e. $\mathcal{O}_1 = \mathcal{L}_{content}$. We observe that, the simple image interpolation (green curve) achieves better results than our method (red curve) when approaching near point C, that is, near the objective $\mathcal{O}_1$. The main reason for that, is that the main network was trained for style transfer, and the ability of the tuning blocks to "Turn the table upside down" and produce image with very little style, is limited. Last, we show that using Dynamic-Net with a smaller range between the objectives, $B \rightarrow C$, (blue curve) outperform both methods and approximate the fixed nets accurately.

6. Conclusions

We propose Dynamic-Net a novel two phase training framework that allow traversing the objective space at inference time without re-training the model. We have shown its broad applicability on variety vision tasks: style transfer, face generation and image completion. In all application we showed that our method allow easy and intuitive control of the objective trade-off. This work is a first step in providing a model that is not limited to a specific static working point – a dynamic model. Future work include bringing the dynamic concept to other application and expend it to other objective spaces.

In the supplementary we present additional results and provide implementation details.

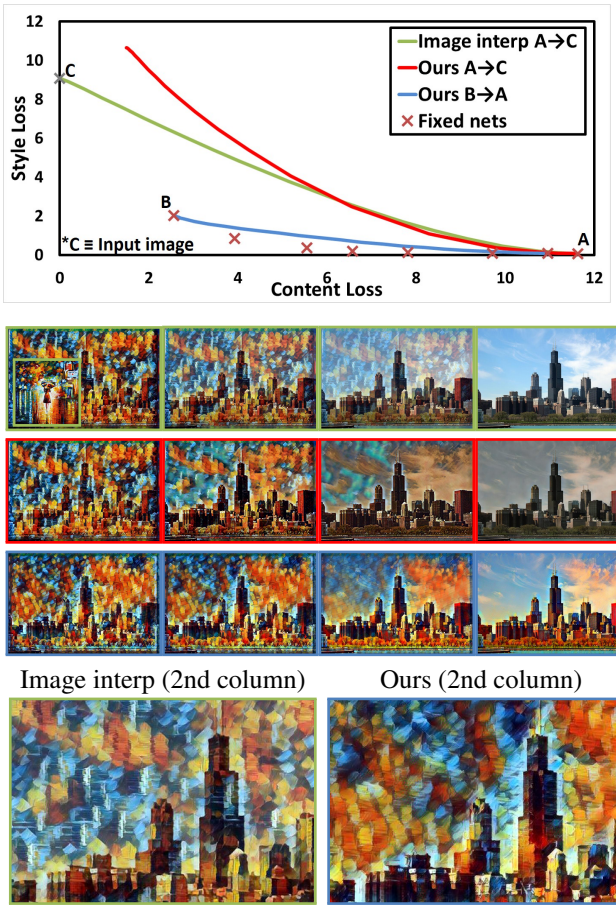


Figure 8: **Failure Case: Top**: Each curve corresponds to a different setting: (green) image-space interpolation between fixed net (A) and the input image. (red) Dynamic-Net results with $\mathcal{O}_0 = \mathcal{O}_A$ ($\lambda = 10^6$) and $\mathcal{O}_1 = \mathcal{O}_C$. (blue) Dynamic-Net results with $\mathcal{O}_0 = \mathcal{O}_B$ ($\lambda = 10^4$) and $\mathcal{O}_1 = \mathcal{O}_A$. **Bottom**: Example images, the box color correspond to the curve color. Training with medium objective range ($B \rightarrow A$) achieved great results, however, increasing the range too much, i.e. $A \rightarrow C$, weaken the results quality.

Acknowledgements

This research was supported by the Israel Science Foundation under Grant 1089/16 and by the Ollendorf foundation.

References

[1] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. In *ICML*, 2013. 2, 3

[2] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *CVPR*, 2018. 1, 2

[3] Ying-Cong Chen, Huaijia Lin, Michelle Shu, Ruiyu Li, Xin Tao, Xiaoyong Shen, Yangang Ye, and Jiaya Jia. Facelet-bank for fast portrait manipulation. In *CVPR*, 2018. 2

[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 2018. 2

[5] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *Proc. of ICLR*, 2, 2017. 5

[6] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 1, 2, 4

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2

[9] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519, 2017. 5

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 1, 2, 4, 8

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 1, 2, 4

[12] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 2017. 1

[13] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 2

[14] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE TPAMI*, 2017. 2

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 4

[16] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018. 1

[17] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 7

[18] Roey Mechrez, Itamar Talmi, Firas Shama, and Lihi Zelnik-Manor. Maintaining natural image statistics with the contextual loss. In *ACCV*, 2018. 1

[19] Roey Mechrez, Itamar Talmi, and Lihi Zelnik-Manor. The contextual loss for image transformation with non-aligned data. In *ECCV*, 2018. 1

[20] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016. 2

[21] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 7

[22] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *ICLR*, 2018. 2

[23] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 2

[24] Wei Shen and Rujie Liu. Learning residual images for face attribute manipulation. In *CVPR*. IEEE, 2017. 1

[25] Paul Upchurch, Jacob R Gardner, Geoff Pleiss, Robert Pless, Noah Snavely, Kavita Bala, and Kilian Q Weinberger. Deep feature interpolation for image content changes. In *CVPR*, 2017. 2, 4

[26] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *arXiv preprint arXiv:1805.04487*, 2018. 2